Software Design Document for the Land Information System

Submitted under Task Agreement GSFC-CT-2

Cooperative Agreement Notice (CAN) CAN-00OES-01

Increasing Interoperability and Performance of Grand Challenge Applications in the Earth, Space, Life, and Microgravity Sciences

Version 4.0

Revision history:

Version	*Summary of Changes*	*Date*
1.0	Initial release.	8/13/02
2.2	Release for Milestone H	3/18/03
3.1	Release for Milestone I	9/15/03
3.2	Revision on CT's comments	10/7/03
3.3	Release for Milestone J	9/7/04
4.0	Release for Milestone K	2/11/05

Table of Contents

List of Figures

List of Tables

Acronyms and Terms

ALMA: Assistance for Land-surface Modeling Activities

API: Application Programming Interface

CGI: Common Gateway Interface

CLM: Community Land Model

DODS: Distributed Ocean Data System

ESMF: Earth System Modeling Framework

GrADS: Grid Analysis and Display System

LDAS: Land Data Assimilation System

LIS: Land Information System

MRTG: Multi Router Traffic Grapher

NFS: Network File System

NOAH: National Centers for Environmental Prediction, Oregon State University, United States Air Force, and Office of Hydrology Land Surface Model

PXE: Preboot Execution Environment

RAID: Redundant Array of Inexpensive Disks

SNMP: Simple Network Management Protocol

VIC: Variable Infiltration Capacity Land Surface Model

1 Introduction

This Software Design Document establishes the software design for the Land Information System (LIS). LIS is a project to build a high-resolution, high-performance land surface modeling and data assimilation system to support a wide range of land surface research activities and applications.

This document has been prepared in accordance with the requirements of the Task Agreement GSFC-CT-2 under Cooperative Agreement Notice CAN-00-OES-01 Increasing Interoperability and Performance of Grand Challenge Applications in the Earth, Space, Life, and Microgravity Sciences, funded by NASA's ESTO Computational Technologies (formerly High Performance Computing and Communications) Project.

1.1 Purpose and goals

This document serves as the blueprint for the software development and implementation of the Land Information System (LIS).

The design goals of LIS are near real-time, high-resolution (up to 1km) global land data simulation executed on highly parallel computing platforms, with well defined, standard-conforming interfaces and data structures to interface and inter-operate with other Earth system models, and with flexible and friendly web-based user interfaces.

1.2 Scope

This document covers the design of all the LIS software components for the three-year duration of the LIS project. The document focuses primarily on the implementation of the LIS software on a general-purpose Linux cluster system, and most of the component designs also apply to an SGI Origin 3000 system. This document does not cover design for other hardware/software platforms.

Specifically, this design covers the following aspects of LIS:

- Realistic land surface modeling. LIS will simulate the global land surface variables using various land surface models, driven by atmospheric "forcing data" (e.g., precipitation, radiation, wind speed, temperature, humidity) from various sources.
- High performance computing. LIS will perform high-performance, parallel computing for near real-time, high-resolution land surface modeling research and operations.
- Efficient data management. The high-resolution land surface simulation will produce a huge data throughput, and LIS will retrieve, store, interpolate, re-project, sub-set, and backup the input and output data efficiently.

- Usability. LIS will provide intuitive web-based interfaces to users with varying levels of access to LIS data and system resources, and enforce user security policies.
- Interoperable and portable computing. LIS will incorporate the ALMA (Assistance for Land surface Modeling Activities) and ESMF (Earth System Modeling Framework) standards to facilitate inter-operation with other Earth system models. In order to demonstrate portability of LIS, the land surface modeling component will be implemented on a custom-designed Linux cluster and an SGI Origin 3000.

2.0 Land Surface Modeling and Data Assimilation

In general, land surface modeling seeks to predict the terrestrial water, energy and biogeochemical processes by solving the governing equations of the soil-vegetation-snowpack medium. Land surface data assimilation seeks to synthesize data and land surface models to improve our ability to predict and understand these processes. The ability to predict terrestrial water, energy and biogeochemical processes is critical for applications in weather and climate prediction, agricultural forecasting, water resources management, hazard mitigation and mobility assessment.

In order to predict water, energy and biogeochemical processes using (typically 1-D vertical) partial differential equations, land surface models require three types of inputs: 1) initial conditions, which describe the initial state of land surface; 2) boundary conditions, which describe both the upper (atmospheric) fluxes or states also known as "forcings" and the lower (soil) fluxes or states; and 3) parameters, which are a function of soil, vegetation, topography, etc., and are used to solve the governing equations.

The proposed LIS framework will include various components that facilitate global land surface modeling within a data assimilation system framework. The main software components of the system are:

- LIS driver: is a software system that is derived from the Land Data Assimilation System (LDAS) that integrates the use of land surface models in a data assimilation framework.
- Land surface Models: LIS will include 3 different land surface models, namely, CLM, NOAH, and VIC.

These components are explained in detail in the following sections.

2.1 LIS driver

The LIS driver that controls the execution of different land models is derived from LDAS. LDAS is a model control and input/output system (consisting of a number of subroutines, modules written in Fortran 90 source code) that drives multiple offline one-dimensional land surface models (LSMs). The one-dimensional LSMs such as CLM and NOAH, which are subroutines of LDAS, apply the governing equations of the physical processes of the soil-vegetation-snowpack medium. These land surface models aim to characterize the transfer of mass, energy, and momentum between a vegetated surface and the atmosphere. When there are multiple vegetation types inside a grid box, the grid

box is further divided into "tiles", with each tile representing a specific vegetation type within the grid box, in order to simulate sub-grid scale variability.

LDAS makes use of various satellite and ground based observation systems within a land data assimilation framework to produce optimal output fields of land surface states and fluxes. The LSM predictions are greatly improved through the use of a data assimilation environment such as the one provided by LDAS. In addition to being forced with real time output from numerical prediction models and satellite and radar precipitation measurements, LDAS derives model parameters from existing topography, vegetation and soil coverages. The model results are aggregated to various temporal and spatial scales, e.g., 3 hourly, 0.25 deg x 0.25 deg. The LDAS driver was used in the baselining results presented for Milestone E. The LIS driver used for demonstrating code improvements for Milestone H was developed by adopting the core LDAS driver and implementing code improvements for enhancing performance. The structure of LDAS driver was also redesigned using object oriented principles, providing adaptable interfaces for ease of code development and extensibility. Details of the LIS driver is presented in the interoperability document and the code improvements are presented in the code improvements documents for Milestone F. The LIS driver was modified to run on the LIS cluster and to include the VIC code for Milestone I.

The execution of LIS driver starts with reading in the user specifications. The user selects the model domain and spatial resolution, the duration and timestep of the run, the land surface model, the type of forcing from a list of model and observation based data sources, the number of ``tiles'' per grid square (described below), the soil parameterization scheme, reading and writing of restart files, output specifications, and the functioning of several other enhancements including elevation correction and data assimilation.

The system then reads the vegetation information and assigns subgrid tiles on which to run the one-dimensional simulations. The LIS driver runs its 1-D land models on vegetation-based "tiles" to simulate variability below the scale of the model grid squares. A tile is not tied to a specific location within the grid square. Each tile represents the area covered by a given vegetation type.

Memory is dynamically allocated to the global variables, many of which exist within Fortran 90 modules. The model parameters are read and computed next. The time loop begins and forcing data is read, time/space interpolation is computed and modified as necessary. Forcing data is used to specify boundary conditions to the land surface model. The LSMs in the LIS driver are driven by atmospheric forcing data such as precipitation, radiation, wind speed, temperature, humidity, etc., from various sources. The LIS driver applies spatial interpolation to convert forcing data to the appropriate resolution required by the model. Since the forcing data is read in at certain regular intervals, the LIS driver also temporally interpolates time average or instantaneous data to that needed by the model at the current timestep. The selected model is run for a vector of ``tiles'', intermediate information is stored in modular arrays, and output and restart files are written at the specified output interval.

2.2 Community Land Model (CLM)

CLM (Community Land Model) is a 1-D land surface model, written in Fortran 90, developed by a grass-roots collaboration of scientists who have an interest in making a general land model available for public use. LIS currently uses CLM version 2.0. CLM version 2.0 was released in May 2002. The source code for CLM 2.0 is freely available from the National Center for Atmospheric Research (NCAR) (http://www.cgd.ucar.edu/tss/clm/). The CLM is used as the land model for the Community Climate System Model (CCSM) (http://www.ccsm.ucar.edu/), which includes the Community Atmosphere Model (CAM) (http://www.cgd.ucar.edu/cms/). CLM is executed with all forcing, parameters, dimensioning, output routines, and coupling performed by an external driver of the user's design (in this case done by LDAS). CLM requires pre-processed data such as the land surface type, soil and vegetation parameters, model initialization, and atmospheric boundary conditions as input. The model applies finite-difference spatial discretization methods and a fully implicit time-integration scheme to numerically integrate the governing equations. The model subroutines apply the governing equations of the physical processes of the soil-vegetation-snowpack medium, including the surface energy balance equation, Richards' (1931) equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Collatz et al. (1991) formulation for the conductance of canopy transpiration.

2.3 The Community NOAH Land Surface Model

The community NOAH Land Surface Model is a stand-alone, uncoupled, 1-D column model freely available at the National Centers for Environmental Prediction (NCEP; ftp://ftp.ncep.noaa.gov/pub/gcp/ldas/noahlsm/). The name is an acronym representing the various developers of the model (N: NCEP; O: Oregon State University, Dept. of Atmospheric Sciences; A: Air Force (both AFWA and AFRL - formerly AFGL, PL); and H: Hydrologic Research Lab - NWS (now Office of Hydrologic Dev -- OHD)). NOAH can be executed in either coupled or uncoupled mode. It has been coupled with the operational NCEP mesoscale Eta model (Chen et al., 1997) and its companion Eta Data Assimilation System (EDAS) (Rogers et al., 1996), and the NCEP Global Forecast System (GFS) and its companion Global Data Assimilation System (GDAS). When NOAH is executed in uncoupled mode, near-surface atmospheric forcing data (e.g., precipitation, radiation, wind speed, temperature, humidity) is required as input. NOAH simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snowpack depth, snowpack water equivalent, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance. The model applies finite-difference spatial discretization methods and a Crank-Nicholson time-integration scheme to numerically integrate the governing equations of the physical processes of the soil vegetation-snowpack medium, including the surface energy balance equation, Richards' (1931) equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Jarvis (1976) equation for the conductance of canopy transpiration.

2.4 Variable Infiltration Capacity (VIC) Model

Variable Infiltration Capacity (VIC) model is a macroscale hydrologic model, written in C, being developed at the University of Washington and Princeton University. The VIC code repository along with the model description and source code documentation is publicly available at http://hydrology.princeton.edu/research/lis/index.html. VIC is used in macroscopic land use models such as SEA - BASINS (http://boto.ocean.washington.edu/seasia/intro.htm). VIC is a semi-distributed, grid-based hydrological model, which parameterizes the dominant hydrometeorological processes taking place at the land surface - atmospheric interface. The execution of VIC model requires preprocessed data such as precipitation, temperature, meteorological forcing, soil and vegetation parameters, etc. as input. The model uses three soil layers and one vegetation layer with energy and moisture fluxes exchanged between the layers. The VIC model represents surface and subsurface hydrologic processes on a spatially distributed (grid cell) basis. Partitioning grid cell areas to different vegetation classes can approximate sub-grid scale variation in vegetation characteristics. VIC models the processes governing the flux and storage of water and heat in each cell-sized system of vegetation and soil structure. The water balance portion of VIC is based on three concepts:

1) Division of grid-cell into fraction sub-grid vegetation coverage.
2) The variable infiltration curve for rainfall/runoff partitioning at the land surface.
3) A baseflow/deep soil moisture curve for lateral baseflow.

Water balance calculations are preformed at three soil layers and within a vegetation canopy. An energy balance is calculated at the land surface. A full description of algorithms in VIC can be found in the references listed at the VIC website.

3 LIS software architecture

This section describes the software architecture of the components of LIS. The proposed LIS framework will have the following functional components: (1) A system for high resolution global land data assimilation system, involving several land surface models, and land data assimilation technologies. (2) A web-based user interface that accesses data mining, numerical modeling and visualization tools. To facilitate these features, LIS will integrate the use of various software systems such as LDAS, land surface models, GrADS – DODS, etc. LIS is also expected to act as a framework that enables the land surface modeling community to define new standards and also to assist in the definition and demonstration of the ESMF. As a result, the design of LIS will also feature the incorporation of new standards and specifications such as ALMA and ESMF.

Figure 1 shows the initial LDAS software architecture. As mentioned earlier, the baselined version of LDAS includes CLM and NOAH land surface models. VIC land surface model will be incorporated in the Milestone I version of LDAS and LIS.

Figure 2 presents the LIS software architecture. It can be noticed that LIS will be built upon the existing LDAS, with new components and expanded functionalities for the support of parallel processing, GrADS-DODS server-based data management, ALMA

and ESMF-compliance, web-based user interfaces, and system management of a Linux cluster platform

The function of LIS dictates a highly modular system design and requires all the modules, or components, to work together smoothly and reliably. Figure 2 shows an overview of the LIS software architecture and its components, and their interactions. LIS will continuously take in relevant atmospheric observational data, and will subsequently use it to force the land surface models, and the land surface simulation is carried out in a highly parallel fashion. Meanwhile the large amount of output data will be efficiently managed to facilitate reliable and easy access. Moreover, LDAS, its interface to the three land models (CLM, NOAH, and VIC), and its input/output modules, will be partially compliant with ESMF, while the output data variables and formats, and the variables passed between LDAS and the three land models, will follow ALMA specification. Finally, LIS also has software components to manage the parallel job processing and monitor hardware status and manage them to ensure sustained high performance output and high availability in the Linux cluster environment. Following is a list of LIS software components:

- Land surface modeling: LDAS and the three land models – CLM, NOAH and VIC. LDAS can be configured to run one, two or all the three land models at the same time.
- Parallel processing: implementation of the parallelization scheme.
- GrADS-DODS server
- Data retrieving
- System monitoring: only applies to the LIS cluster environment.

By the use of modular programming and by conforming to well established standards such as ALMA and ESMF, LIS is expected to provide a flexible, extensible framework to land surface modelers and researchers. A more detailed discussion of the ESMF interfaces is located in Section 4 of the Interface Design for Interoperability.

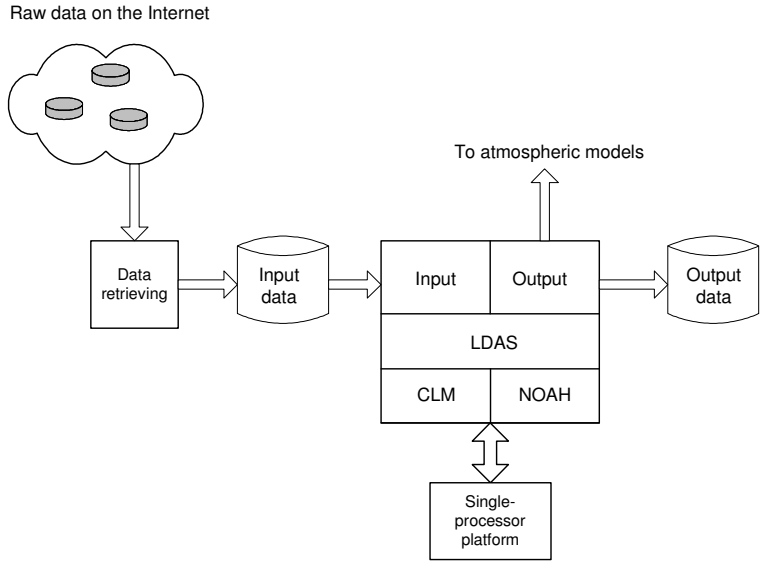

Figure 1: Current Land Data Assimilation System (LDAS) structure. It uses CLM and NOAH land

models.

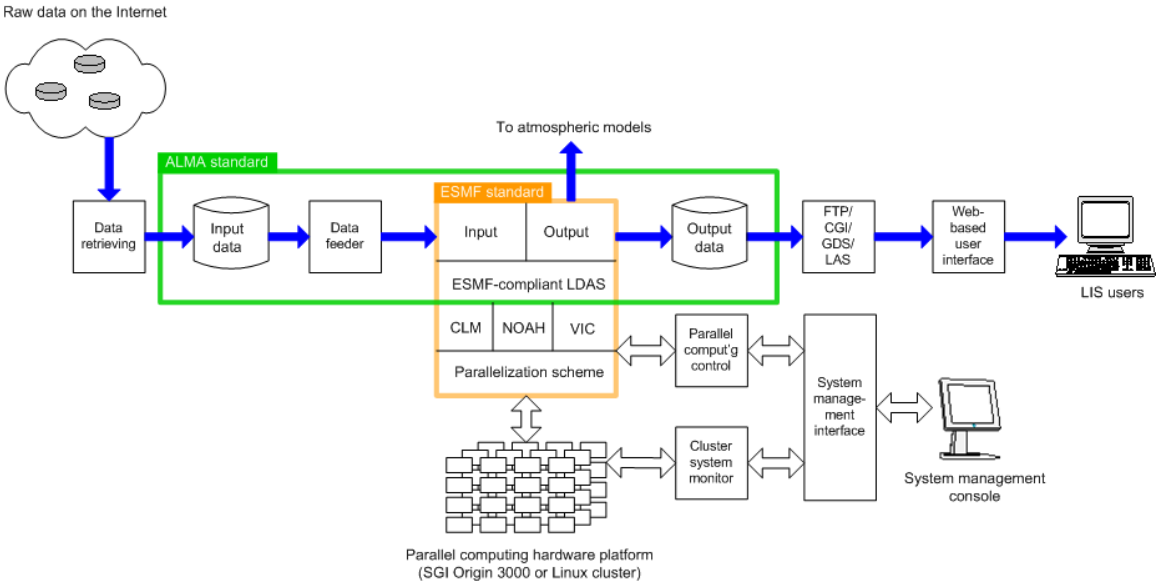

Figure 2: Overview of LIS software architecture and its components designed for LIS cluster. A subset of the components, the LDAS and parallel computing implementation, will also be tested on SGI Origin platforms.

3.1 Software data structures

This section describes the internal software data structures in LIS. As described earlier, the main component that drives different LSMs is the LIS driver. The one-dimensional land surface models such as CLM, NOAH, and VIC are included as subroutines of the LIS driver. LDAS, CLM, and NOAH are written in Fortran 90, and VIC is written in C. The LIS driver code is designed in a modular fashion, with a number of modules used to encapsulate data as well as parameters that are used to solve different governing equations. Please refer to the LIS code documentation http://lis.gsfc.nasa.gov/source/ for a detailed description of the source code.

Figure 3 shows the organization of the main modules in the LIS driver.

Inheritance can be defined as the sharing of structure and behavior among classes in a hierarchical relationship. Although F90 does not directly support inheritance, it can be emulated using software constructs (ref: Decyk, V. K., Norton, C. D., and Szymanski, B. K. "How to Express C++ concepts in Fortran 90". http://exodus.physics.ucla.edu/Fortran95/ExpressC++.pdf)

For example, inheritance in LIS is simulated by the lsm_module that captures the behavior of a land surface model. It also provides a hierarchical structure to all LSMs. The "abstract" interfaces in lsm_module (encapsulating the main behavior associated with the operation of a LSM) need to be implemented by all LSMs in LIS. As a standard for land surface model parameters, input data, and output evolves, this structure is further

expected to allow code sharing among different LSMs, all of them using common routines for intialization, setup, output etc.

A more detailed description of the design is presented in the interoperability design document.

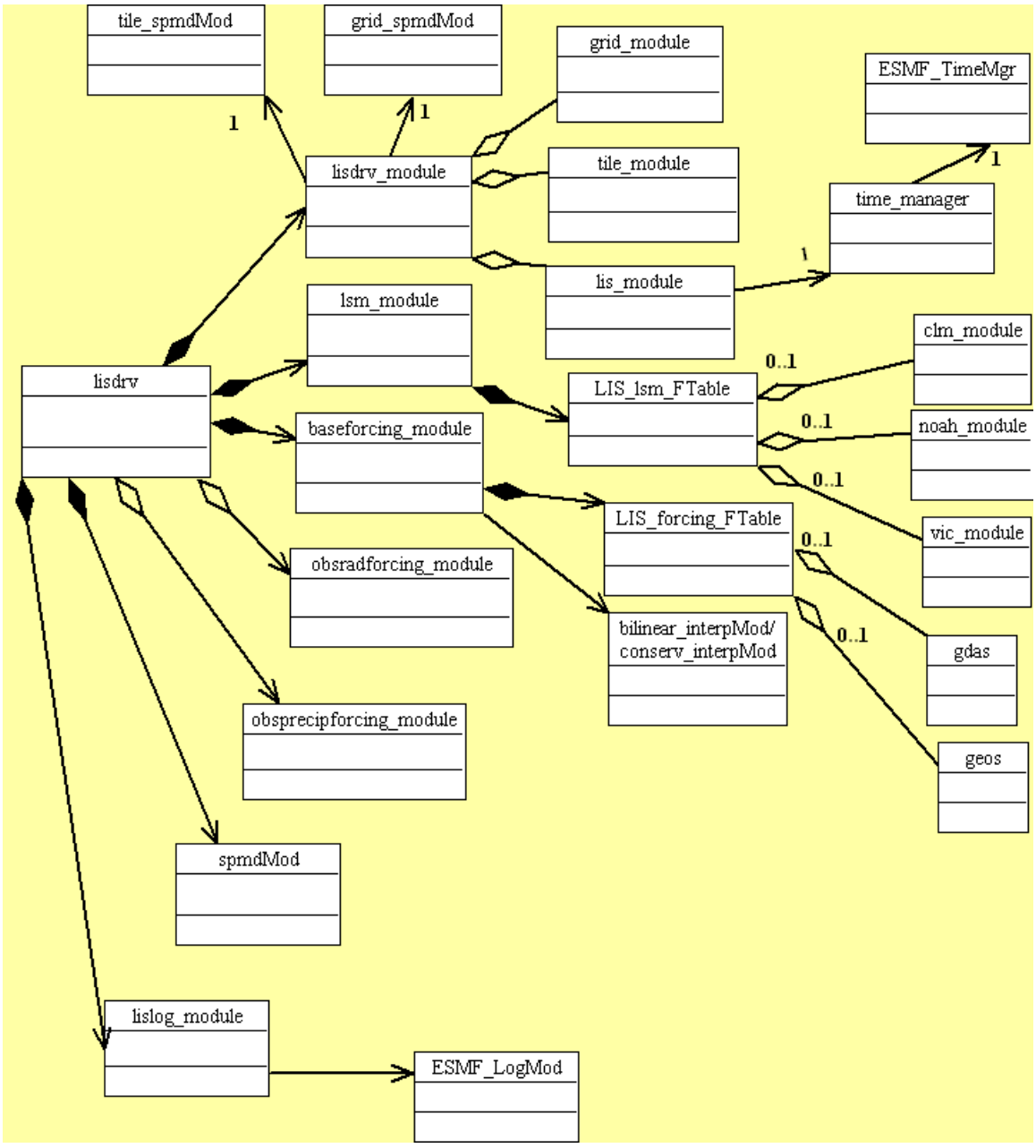

Figure 3: Structure of modules in the LIS driver

A brief description of the modules are presented below:

LIS driver Modules

lisdrv: is the main program in LIS driver. It controls the overall execution, and delegates tasks to the appropriated modules

lisdrv_module: This module contains the driver routines that control program execution, controlling of time, etc.

lsm_module: This module provides an abstraction of a land surface model, defining the interfaces and subroutines that are required for the execution of an LSM. The interfaces in this module need to be extended for incorporation of a new LSM into LIS.

baseforcing_module: Similar to **lsm_module**, this module captures the behavior associated with introducing a new forcing scheme.

grid_module: This module is an abstract representation of a "grid" used in the LIS driver. The module includes non model specific parameters such as lat/lon and input/output forcing variables.

tile_module : This module is a representation of the "tile" described in section 2.1 that is used to simulate sub-grid scale variability. This module includes specification of non-model specific tile variables, such as lat/lon of tile, row/column of tile and properties associated with a tile.

lis_module : This module specifies the variables used in the LIS driver such as the model domain specifications, type of land surface model used, type of forcing, specification of source files, etc. It does not include specification of tile space or grid space variables. This module is used by the main driver and subroutines that perform non-model specific computations such as spatial/temporal interpolation.

obsradforcing_module: This module contains interfaces and subroutines that control the incorporation of observed radiation forcing.

obsprecipforcing_module: This module contains the interfaces and subroutines that control the incorporation of observed precipitation forcing.

spmdMod: This module contains MPI routines for initialization.

time_manager: This module contains variables and routines for the control of time. **time_manager** provides methods that eventually call the ESMF time manager.

tile_spmdMod: This module contains routines for domain decomposition in tile space.

grid_spmdMod: This module contains routines for domain decomposition on the grid domain.

bilinear_interpMod and **conserv_interpMod:** These modules contains routines for calculating parameters required for spatial interpolation

agrmetdomain_module: This module contains routines for calculating parameters required for spatial interpolation for AGRMET radiation forcing data

cmapdomain_module: This module contains routines for calculating parameters required for spatial interpolation for cmap precipitation forcing data

LSM specific modules

The LSMs included in the LIS driver implements the interfaces and routines defined in **lsm_module.** Currently the LIS driver includes NOAH and CLM models. The main Modules in these models are described below.

clmtype: This module contains the definition of variables associated with CLM.
clm_varcon: Defines the constants associated with CLM model execution.
clm_varctl : Defines the run control variables associated with CLM model execution.

noah_module : This module specifies one-dimensional NOAH land driver variable specification. It includes NOAH state parameters, output variables, etc.

VIC structures

VIC includes a number of structures that are used to encapsulate model options, forcing parameters, global simulation parameters, soil and vegetation parameters, etc. The main structures are:

option_struct : This structure is used to store model options.
global_param_struct : This structure is used to store the global parameters defined for the current simulation.
soil_con_struct : This structure is used to store the constant variables for the soil in the current grid cell.
veg_con_struct : This structure is used to store all constant parameters for the vegetation types in the current grid cell.
atmos_data_struct : This structure is used to store the meterological forcing data for each time step.
cell_data_struct : This structure is used to store the grid cell specific variables, not included in the vegetation structures.
energy_bal_struct : This structure is used to store all variables used to compute the energy balance and soil thermal fluxes.
snow_data_struct: This structure is used to store all variables used by the snow accumulation and ablation algorithm, and the snow interception algorithm.

4 Hardware Platforms for LIS

This section describes the hardware operational platforms intended for LIS. The SGI Origin 3000 will be used to implement and demonstrate only the high resolution, parallel, global land surface modeling and data assimilation components (LDAS/CLM/NOAH/VIC) of LIS. The fully operational LIS (with user interfaces and visualization components such as GrADS - DODS) will be demonstrated on a custom designed Linux cluster. The following section describes the hardware design of the cluster.

4.1 LIS cluster architecture

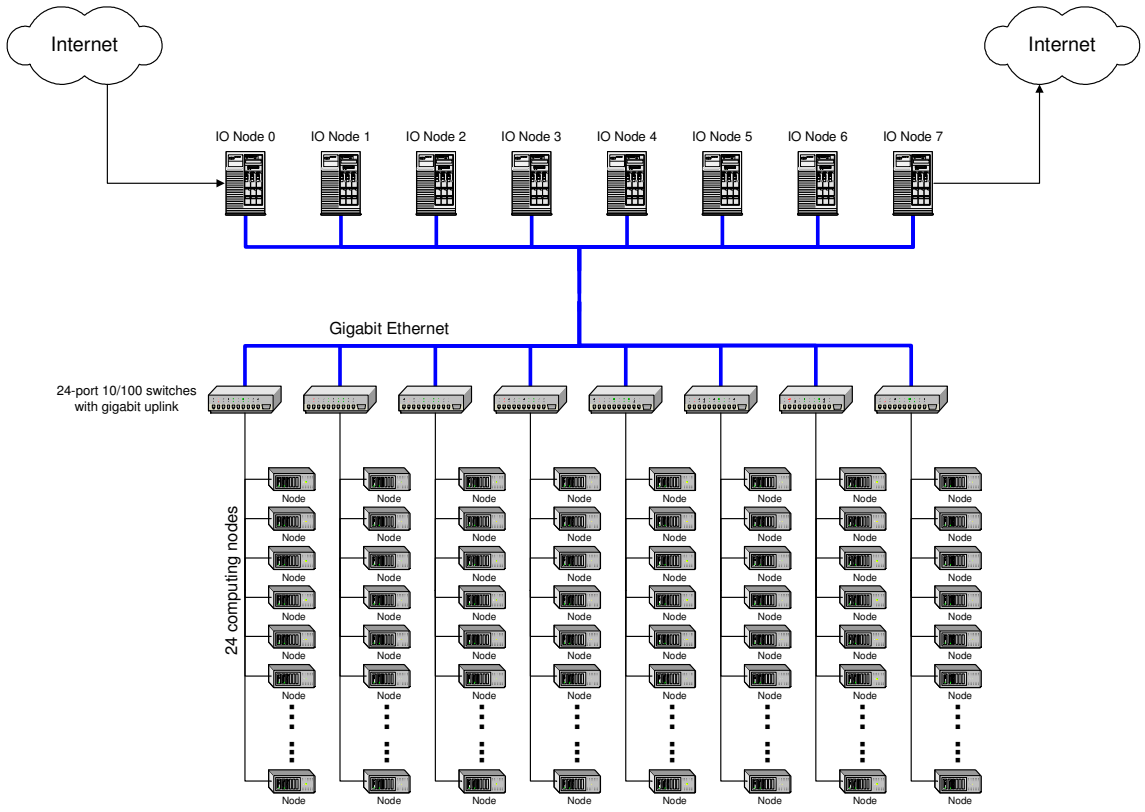

Figure 4: The physical architecture of the LIS Linux cluster. The cluster has 8 IO nodes and 192 compute nodes. Each IO node has dual Athlon CPUs, 2GB RAM and Gigabit NICs, and each compute node has a single Athlon CPU , 512MB RAM and a Fast Ethernet NIC.

Figure 4 shows the physical architecture of the LIS Linux cluster. The cluster consists of 192 computing nodes. The cluster also includes 8 IO (input – output) nodes, specifically to handle the huge data management requirements. These nodes are interconnected with 8 Ethernet switches.

The 192 computing nodes are divided into 8 sub-clusters, with 24 nodes in each sub-cluster, interconnected with fast Ethernet via one of the 8 24-port fast Ethernet switches. Each switch also has two gigabit ports to connect the 8 IO nodes and the other switches.

The use of 8 sub-clusters and 8 IO nodes is mainly for the segregation of network traffic resulting from non-local file IO operations, and for the spreading of data storage so each IO node does not have to deal with single big files. So in average each IO node will only need to serve the IO requests of 24 computing nodes, and only store 1/8 of the output information, which makes the output volume manageable.

4.2 System Monitoring

The system monitoring component is responsible for monitoring, maintaining and administering the LIS system on the Linux cluster to ensure its reliable operation and optimal performance output.

We categorize the system management function into four levels: hardware level, interconnect level, operating system level and application software level. For the SGI Origin 3000 platform, we are not involved in the management of the hardware and interconnect levels. But for the Linux cluster, the hardware and interconnect level management is our responsibility and is critical to the overall stability and performance of the LIS system.

The hardware level system management involves power-up and shutdown of the nodes, booting strategy and hardware status monitoring. Interconnect level management requires the monitoring of the link status of the network nodes, bandwidth usage and traffic statistics. Operating system level management takes care of system resource usages, such as CPU, memory and disk space usage. Application level management oversees the progress of the LIS jobs, configures different runs, analyze performance bottlenecks, and obtain performance profiles for fine-tuning. Dynamic error and diagnostic logs will be maintained for LDAS and the land surface models during the operation of LIS. The diagnostic logs will be available to the end users.

4.2.1 Hardware monitoring data

The following table summarizes the system data of various levels the management subsystem is designed to collect and analyze.

Table 1: Hardware monitoring and management data collection

LIS Cluster System Monitoring and Management Data		
Category	***Data Items***	***Update frequency***
Application level	Overall cpu/mem of each process	1min
	Overall progress of whole job	2min
	Progress of each process	1min
	Timing of each module	sampled, off-line
	Memory usage of each module	sampled, off-line
Operating system level	Total memory usage & biggest user	2min
	Total CPU usage & biggest user	2min
	Total disk space usage	2min
	System up-time and running procs	2min
Interconnect level	Bandwidth usage of each node	2min
	Bandwidth usage of switches	2min
	Latency measurements	2min
	Packet drops measurements	2min
Hardware level	Fan speeds	10min
	Chasis temperature	10min
	Power supplies voltage	10min

4.2.2 Architecture and implementation

The variety of system variables and management duties requires us to design a management system with modules performing individual and well-defined tasks. Figure 5 shows the structured design of the system management functionalities for the Linux cluster platform. We will not implement such a system on SGI because the SGI platform is not under our management control.

On the hardware level, we will design scripts to take advantage of the “Wake-on-Lan” technology for powering up the nodes smoothly in a well-defined pattern. The nodes will be able to boot across the network with the PXE technology, as well as from the local disk, to centralize system software management. After booting, each node’s hardware parameters, such as CPU temperature, cooling fan speeds and power supply voltages, will be collected by kernel modules called “lm-sensors”, and sent to the central management station with web-based display with automatic updates.

On the interconnect level, we will use SNMP protocol as the underlying data collection and management mechanism, interfaced with MRTG for web-based display of network statistics. Additional network data can also be collected by Big Brother system and network monitor, also with web output.

On the operating system level, we will use SNMP and various OS shell commands and utilities to collect system data, and use MRTG and Big Brother as the interface.

On the application level, we will develop CGI scripts, interfaced with OS commands and utilities, to provide a web-interface for the monitoring and control of LIS jobs and processes. Standard performance profiling and debugging tools will be used off-line to analyze sample runs for trouble-shooting and performance fine-tuning.

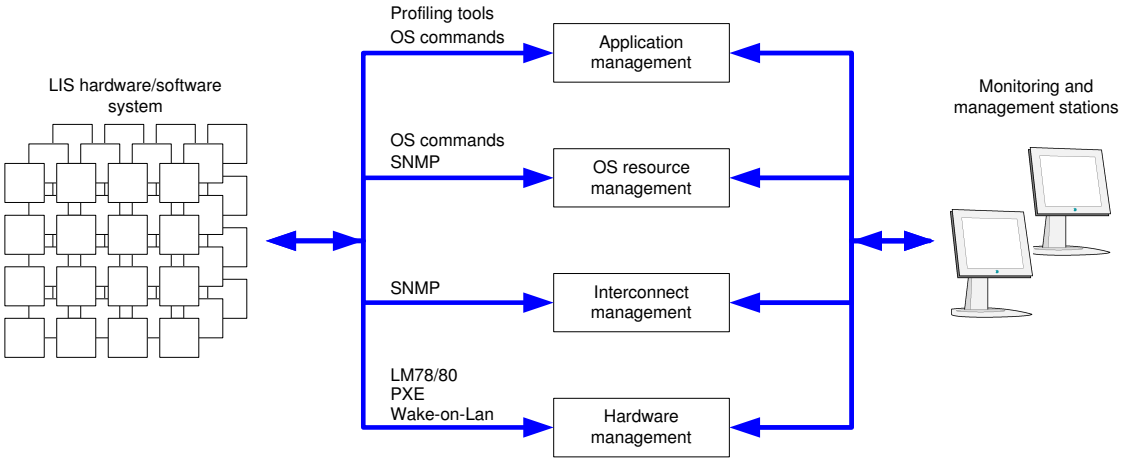

Figure 5: LIS system monitoring and management architecture for the LIS Linux cluster. This system will not be implemented on SGI since it is not under our control.

5 High performance computing in LIS

Accurate initialization of the land surface moisture, carbon, and energy stores in a fully coupled climate system is critical for meteorological and hydrological prediction. Information about land surface processes is also of critical importance to real-world applications such as agricultural production, water resource management, flood prediction, water supply, etc. The development of LDAS has been motivated by the need for a system that facilitates land surface modeling with an assimilation system to incorporate model derived and remotely sensed data. LDAS system has been successfully used in simulations for North America at 1/8 degree resolution in both real time and long term (50 years) retrospective simulations. However, to truly address the land surface initialization and climate prediction problem, LDAS needs to be implemented globally at high resolution (1km). The computational and resource requirements increase significantly for global modeling at such high resolutions. The proposed LIS system will aim to make use of scalable computing technologies to meet the challenges posed by the global, high-resolution land surface modeling.

5.1 Parallel processing in land surface modeling

Parallel computing is a powerful programming paradigm to deal with computationally intractable problems. The notion behind parallel programs is to divide the tasks at hand into a number of subtasks and solve them simultaneously using different processors. As a result, a parallel system can improve the performance of the code considerably.

The land surface modeling component in LIS is designed to perform high-performance, parallel simulation of global, regional or local land surface processes with initially three land surface models: the CLM model, the NOAH model and the VIC model. Specifically, the land surface modeling component will interact with the data management components to obtain properly formatted input forcing data, and pass the forcing data, along with other static parameters, to the three land surface models through the LIS driver. Each of the land surface models carries out the simulation on a distributed, parallel hardware platform, either a Linux cluster or a SGI Origin 3000. The results are passed to the output component, which interacts with the data management subsystem to handle the output data.

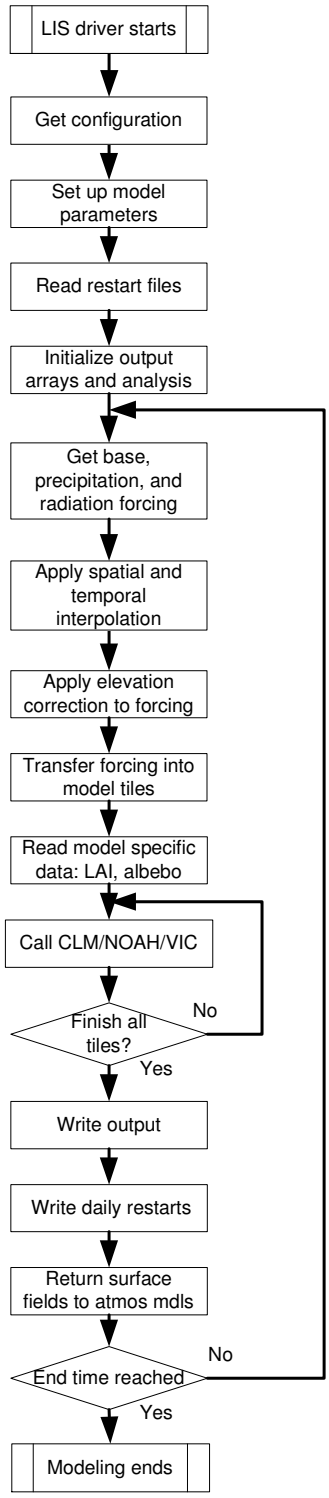

Figure 6: LIS land surface modeling flowchart

As shown in Figure 6, and described in detail in the land surface model documentation, land surface models proceed in a manner similar to other physical models. Modeling proceeds given prior knowledge of the spatial and temporal domains of the simulation, in addition to initial conditions and parameters required to solve the equations of water and energy conservation within that domain. Modeling proceeds according to increments of

time ("time steps", typically 15 minutes), until the ending time is reached and data is written out for future runs and analysis.

The land surface modeling subsystem is designed to be running in parallel, both on a Linux cluster with 200 nodes, and on a SGI Origin 3000 platform with 512 processors. Although the hardware architecture differs greatly between the distributed-memory Linux cluster and the shared-memory SGI Origin 3000, our implementation of the land surface modeling programs will make this architectural difference fairly transparent: On the Linux cluster, each node will run a copy of the land surface modeling process; on the SGI Origin, each CPU will run a copy. Thus we establish a direct correspondence between a node in the Linux cluster and a CPU in the Origin 3000.

Land surface processes have rather weak horizontal coupling on short time and large space scales, which enables highly efficient scaling across massively parallel computational resources. LIS aims to take advantage of this weak horizontal coupling of land surface processes by using a coarse-grained parallelization scheme, which does not require communication between the compute nodes. This design fits well with the distributed memory nature of the Linux cluster architecture.

The parallelization scheme employed in the land surface modeling component in LIS is based on a master slave paradigm, where a master processor performs the initializations, and domain decomposition. The compute nodes perform computations on the decomposed domain. The master processor carries out the global output routines once the compute nodes finish their tasks. The parallel processing component plays a critical role to connect the land surface modeling job to the underlying multi-processor parallel computing hardware platform, in our case, a Linux cluster or an SGI Origin 3000, to achieve the goal of near real-time processing of high-resolution land surface data.

We estimate that at 1km resolution LIS will deal with ~50,000 times more grid points than the 2ºx2.5º resolution. The baselining report from Milestone E estimates that the memory requirements at 1km is in the order of terabytes, which is unmanageable either on the Linux cluster or on the shared memory SGI platforms. The code improvements and redesigns conducted for Milestone F significantly reduced these memory requirements. However, the projected memory requirements from the improved LIS code from Milestone F still estimates approximately 500GB for 1km execution. This makes the simple paradigm, where the master handles the global initializations, intractable. To avoid the bottleneck from this scheme, we plan to redesign the input data flow taking advantage of the GrADS-DODS (GDS) servers' features. GDS provides capabilities for a client to dynamically retrieve subsets of data on a global domain. A GDS Server on a master node will perform the tasks of serving data to the compute nodes. The domain decomposition can be achieved by the compute nodes making requests for data on the domain they are performing the computation, instead of a master processor distributing data to them.

To satisfy the requirements of real-time operation, the job, which includes a grid representation of the global land surface, must be split into smaller pieces and run in parallel. We plan to divide the global surface into 10,000 small land pieces, and with 1km

resolution, each piece would require about 5 times as many computations as the 2ºx2.5º LDAS, and will take a single computing node about 200MB memory to run, and 10 minutes to finish a 1-day simulation, based on the initial performance baselining of LDAS running at both 2ºx2.5º and 0.25ºx0.25º resolutions. The Linux cluster can consume approximately 200 pieces per round, and under ideal conditions, it will take the whole cluster about 50 rounds to finish the whole job. This will take 500 minutes, or about 9 hours, to finish a 1-day simulation of the whole global land surface, which satisfies the real-time requirement with enough extra room. We expect that the timings on the SGI Origin will be comparable to those on the cluster, although memory and disk limitations, some imposed by the queue structure, will likely prohibit effective use of that system for demonstrating LIS in a near-real-time mode. However, we plan to demonstrate the LIS on the SGI Origin system as proof-of-concept.

A compute node's job is to run a copy of the land surface modeling subsystem in its process space, compute a piece of land surface obtained from the IO node, and request another piece of land surface from the IO node as soon as it finishes the current piece, until the IO node refuses to give it any pieces, in which case there are no more land pieces are available and the compute node's job is done. Figure 7 shows the flow chart of the compute node's job handling process.

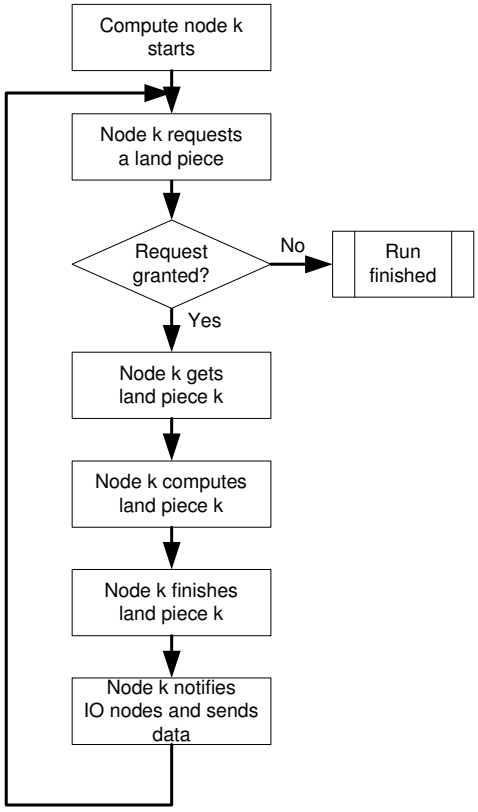

Figure 7: Compute nodes flowchart for parallel computing of land surface modeling. A compute node does not communicate to other compute nodes.

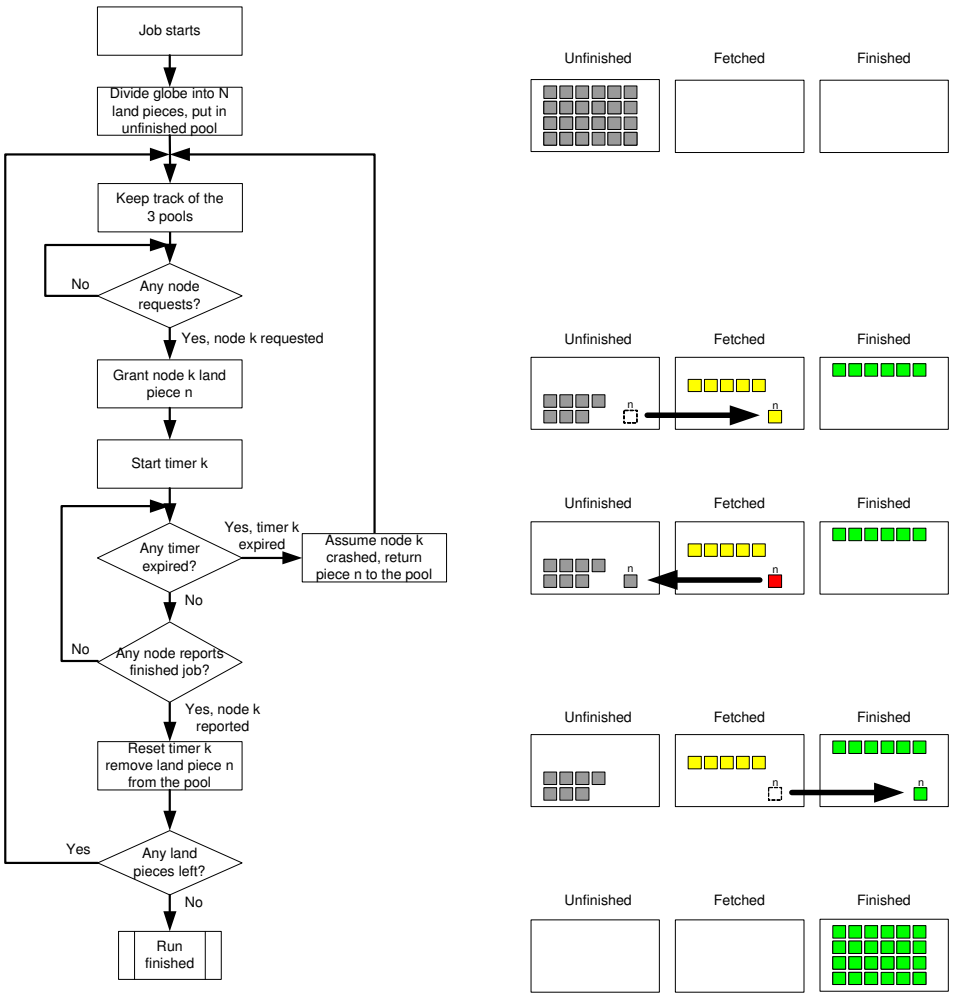

Figure 8: Parallel computing control flowchart (left) and parallelization scheme (right) of a master node.

We propose to use a modified version of the “pool of tasks” scheme for the parallel processing of the land pieces. A pool of tasks paradigm is equivalent to a master – slave programming notion, where a single processor will act as a master node and distribute jobs to the slave (compute) nodes. In the LIS “pool of tasks” design, one of the IO nodes will act as a master node and another IO node will be designated as a backup to it.
The master node will keep three tables on hand when starting the job: table of unfinished-jobs, finished-jobs, and jobs-fetched. At the beginning, the 10,000 land pieces are listed in the "unfinished" table, and each compute node comes to the master to fetch a piece from it, and starts working on it. The master node then moves the fetched jobs to the "jobs-fetched" table, and starts a timer for each fetched job. The timer specification will be based on the existing knowledge of a single execution of a land surface model. When a compute node finishes a job and notifies the master node before the job’s corresponding timer runs out, this piece is regarded a finished job, and the master node moves it from the "fetched" table to the "finished" table. And the compute node goes on to fetch another job until the "unfinished" table is empty. If a fetched job's timer runs out before the compute node reports back, the master node then assumes that that particular compute node must have crashed, and then moves that timed-out job from the "fetched" table back

to the "unfinished" table for other compute nodes to fetch. Figure 8 shows the flowchart (left) of the master node's job handling and scheduling process, and the various status of the three tables (right) the master node uses to keep track of the job progress at different corresponding stages in the flowchart.

To maximize throughput of the system in a parallel environment, load balancing is required to keep the compute nodes busy by efficiently distributing the workload. The use of a "pool of tasks" is effective in achieving automatic load balancing by minimizing the idle times of compute nodes, since the nodes that finish their computations will request more tasks than the ones that require more time for their computations. This automatic, asynchronous scheduling help in keeping the compute nodes busy without having to wait for other node's computations.

As shown in Figure 8, as the land surface modeling process starts, the master node divides the globe into a number of smaller pieces. The inputs required by the land surface models, namely, initial conditions, boundary conditions, and parameters will be provided to the compute nodes before the land surface model run begins. The modeling process can be a fresh initialization (cold start) or a restart from a previously finished run. This process also requires preprocessing of the data such as time/space interpolation. The output from each compute node, after the computation, will be reassembled at the IO nodes.

6 Data Management in LIS

The data management subsystem in LIS is composed of the following functions: input data retrieval from the Internet, data pre-processing and post-processing, data interpolation and sub-setting, output data aggregation, storage, backup and retrieval. It links the other subsystem together, and ensures smooth end-to-end data flow, from the input raw data all the way to the output data satisfying LIS users' various requests. The following sections describe the data flow and volume used in LIS operation, the use of GrADS-DODS server for data management, visualization, the Live Access Server (LAS) server for visualization, etc., and other functions such as data retrieval.

The full data management design is now documented in the separate Data Management Design Document. This section only describes the software architecture of the GrADS-DODS server.

6.1 GrADS-DODS server architecture

GrADS-DODS servers will be employed both to serve the input data to the land surface computing code, and to serve the output to the Internet users. Figure 10 shows the architecture of the GrADS-DODS server. A GrADS-DODS server uses a typical client-server architecture to communicate with the DODS clients. The communication protocol between a client and a server is HTTP. A GrADS-DODS server has the following components: Java servlets contained in the Tomcat servlet container, to handle the client requests and server replies via HTTP protocol; DODS server APIs, to parse the DODS requests and package output data; interface code, to translate the DODS requests into

GrADS calls; and finally, GrADS running in batch mode, to actually process the requests, and perform data-retrieving, sub-setting and processing on the server side.

The LAS Server provides an additional web interface for users to search a data catalog, to visualize data interactively, and to download the data in various formats. LAS uses perl scripts to retrieve the metadata from the LIS output files, and save the metadata in a SQL database system, MySQL. The LAS server and its accompanying database, MySQL, will be running on the same LIS cluster node.

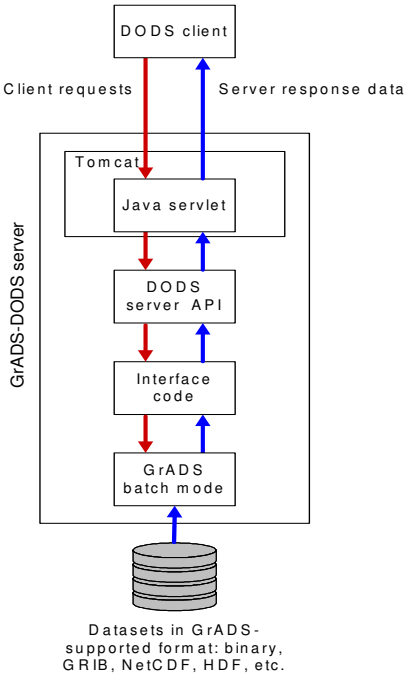

Figure 9: GrADS-DODS server architecture.

7 Interoperability and Community Requirements

Interoperability means the ability of a system to use parts of another system and also provide parts of itself that can be used by other systems to ease the cost of development and foster better interaction between different research groups. Code interoperability is important not only between components within an application, but also between different applications. LIS defines two different types of interoperability: internal and external. Internal interoperability mainly deals with the aspects of making components within LIS interoperable and the external interoperability deals with the interaction of LIS with other related scientific community applications and standards.

7.1 Internal Interoperability

Interoperability within the LIS will allow for the addition of improved sources of input data and land surface models as they become available. As currently designed, the LIS

has three land surface models available for use. The LIS community has identified a number of other land surface models that would be scientifically beneficial. Likewise with the input data, additional sources of input data would be scientifically beneficial if they were available in LIS.

Interoperability within the LIS system is achieved by reorganizing the LIS driver to organize the code and module control into a framework by designing flexible and adaptive interfaces between subsystems. The LIS driver makes use of the advanced features of the Fortran 90 programming language, which are especially useful for object-oriented programming. The design uses object-oriented principles to provide a number of well-defined interfaces or "hook points" for adding additional land surface models and input data sets. These interfaces are implemented by using a number of virtual function tables and the actual delegation of the calls at runtime is done at runtime by resolving the function names from the table. C language allows the capabilities to store functions, table them, and pass them as arguments. F90 allows passing of functions as arguments. By combining both these languages, LIS uses a complete set of operations with function pointers. The LIS driver will provide the land surface modeling community an avenue to easily add additional models or input data through the use of such an extensible system. A more detailed description can be found in Section 3 of the Interface Design for Interoperability for the Land Information System on the LIS web site.

7.2 External Interoperability

The LIS design also needs to be interoperable with frameworks outside of LIS so that the outputs from LIS can be useful to weather and climate models. External interoperability is achieved by adopting the ALMA data exchange convention and by being a partially compliant component of the ESMF. By following the ALMA standard, the LIS land surface modeling system is guaranteed to exchange data with other land surface modeling systems that are also ALMA-compliant.

ESMF compliance will allow us to interact with other Earth system models, such as atmospheric models or climate models with compliant interfaces. ESMF is intended to provide a structured collection of building blocks that can be customized to develop model components. ESMF can be broadly viewed as consisting of an infrastructure of utilities and data structures for building model components and a superstructure for coupling and running them. ESMF provides a utility layer that presents a uniform interface for common system functions. LIS has implemented a number of ESMF utilities including configuration, time management and use of basic ESMF data structures. ESMF also defines a number of guidelines for applications that are intended to be coupled. For gridded components, ESMF provides standard methods for components to be initialized in parallel configurations and destroyed. LIS has also implemented a prototype using these interfaces to demonstrate pseudo-coupling with an atmospheric component.

Figure 10 shows the structure of both internal and external interfaces in LIS. The input and output data in LIS will conform to ALMA data exchange standards. The LIS driver will provide a structured set of interfaces for incorporation of new LSMs. Further, the

LIS driver will provide an ESMF complaint interface and use the ESMF_State to exchange information with other ESMF compliant systems. A more detailed description of interoperability design issues can be found in the Interface Design for Interoperability for the Land Information System at the LIS web site.

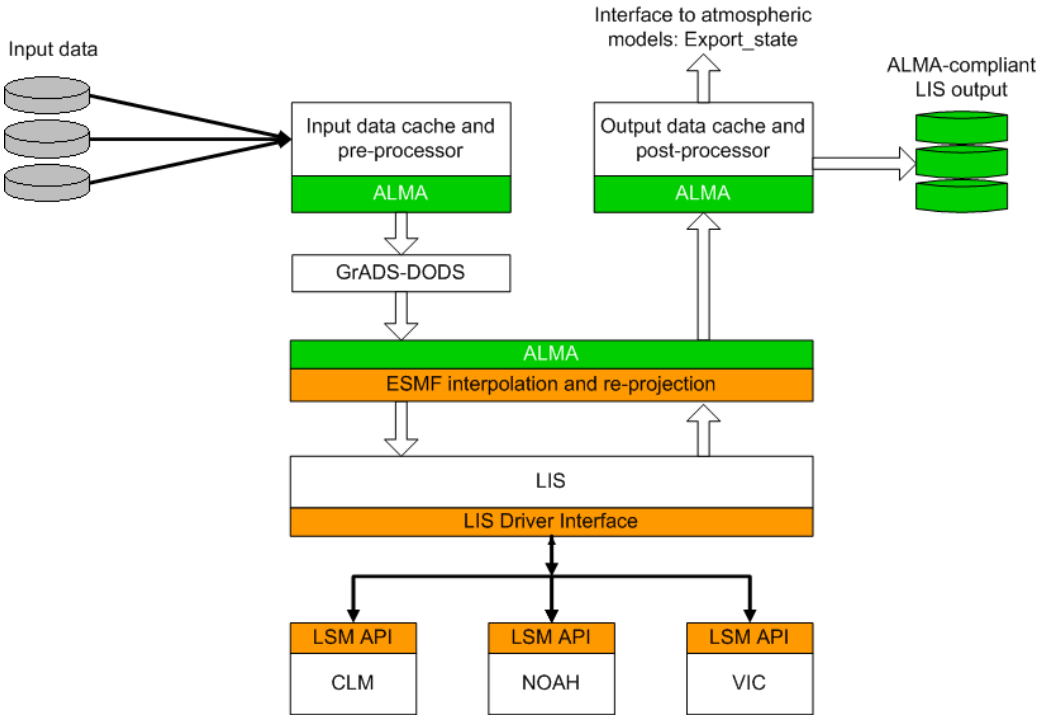

Figure 10 : Interfaces for Interoperability in LIS

8 User interface design

The user interface in LIS is an important component of LIS that will allow the interactive, flexible, use of the LIS hardware and software to users. The LIS user interface is intended to be web-based, and designed to allow for cascading complexity depending on the level of user's need to control the system. The following sections present various facets of the user interface design of LIS.

8.1 User interface components

The user interface subsystem takes a typical multi-tier client-server system architecture. On the client side, a user has three types of client programs to use as the front-end: a web browser, an ftp client program (which can be integrated in a web browser), or a DODS client program. On the server side, a general purpose web server will be used to serve clients with a web browser, and a GrADS-DODS server will be deployed to serve DODS clients, and a FTP server to server ftp clients. Besides these components, CGI scripts and CGI-GrADS gateway scripts will be used as the middleware to perform dynamic

processing based on users' interactive requests sent through web browsers. The following figure shows the user interface architecture design.

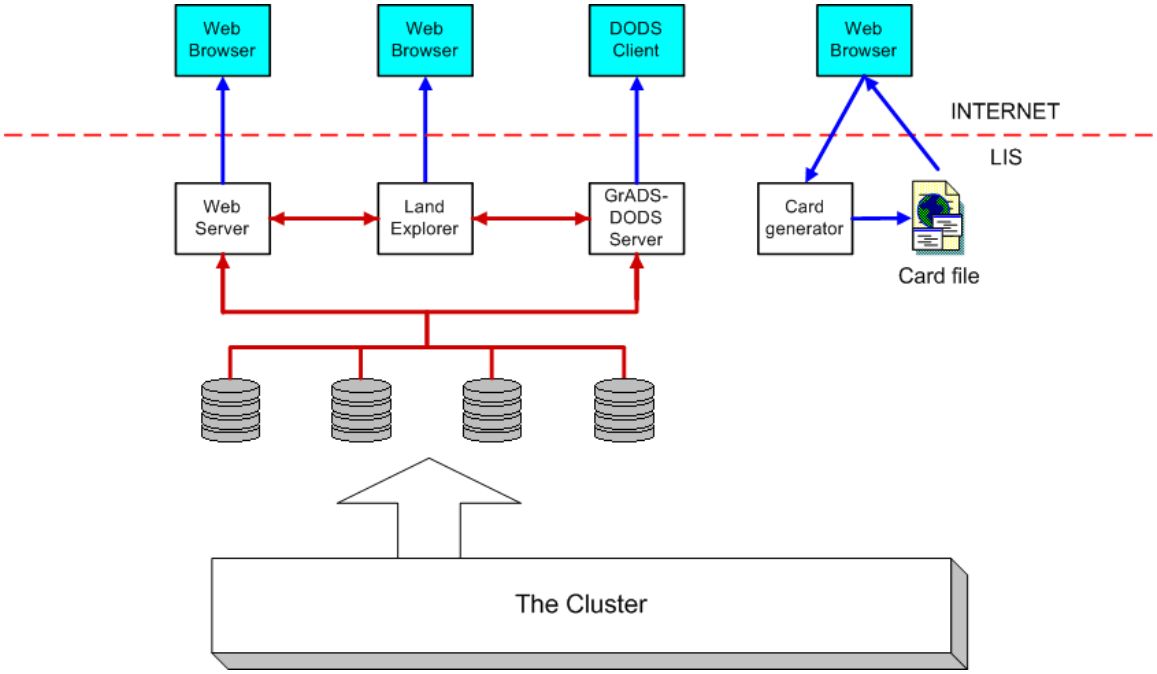

Figure 11: LIS user interface architecture.

.

Additional information on the User Interface Design for the Land Information System can be found at the LIS web site http://lis.gsfc.nasa.gov/documentation/.

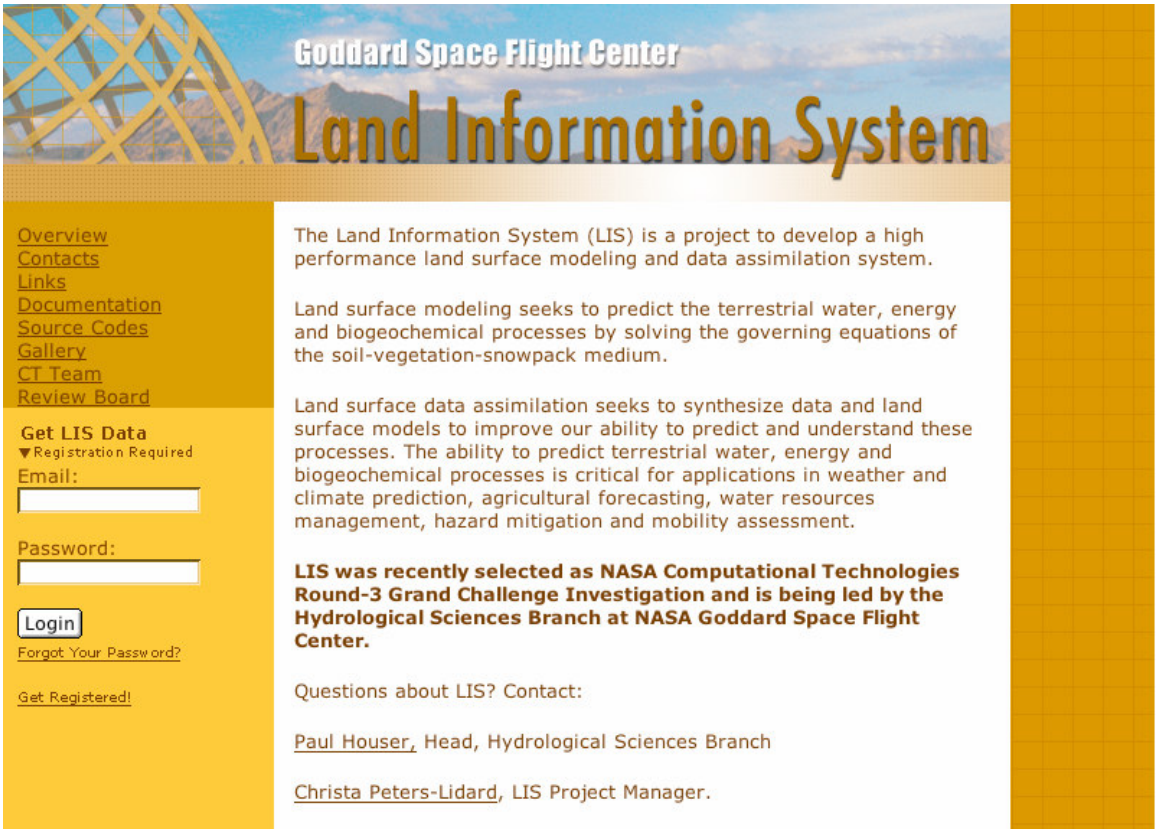

Figure 12: Screenshot of LIS web entry page.

8.2 User Levels

Outside users accessing the LIS are categorized into three levels, associated with different levels of data access and security requirements.

Level 1 users are the general public, who will access the LIS data primarily through a standard web browser. Information provided to this class includes static images and text, and some limited interactive content such as GIF/JPG/PNG images generated on the fly in response to users' regulated web input. The static content, most of which is static html pages, is served via the web server, while the interactive content is generated via a three-tier architecture with server-side GrADS as the image engine and below it the GrADS-DODS server as the data engine to feed the server-side GrADS. This group of users does not have direct access to the data or LIS scientific computing power system, and their usage of system resources is very limited. Therefore, for this class of users we do not enforce any additional authentication or authorization procedures. It is also our intention to facilitate easy access to the data for education and outreach purposes. Figure 14 is a screenshot of the LIS entry page.

Level 2 users have direct access to LIS data, either through our GrADS-DODS server by using a DODS client, or directly through HTTP downloads. The GrADS-DODS server provides the users with the ability and flexibility to get only a sub-set of the data they need. To be authorized as Level 2 users, they will have to register with us first by filling

out web forms, and they will be authenticated using name and password before accessing the data. The GrADS-DODS server will impose a limit on system resource usages. The GrADS-DODS server allows the system administrator to limit the system usage by configuring the following parameters for each authorized address:

Table 2: Configurable GrADS-DODS parameters for access to level 2 users of LIS

Parameter	Description
Subset limit	Sets the maximum size in megabytes of a subset
Generate limit	Sets the maximum size in kilobytes of a generated dataset
Upload limit	Sets the maximum size in kilobytes of an uploaded dataset
Time limit	Sets the maximum time in milliseconds that a dataset generation task is allowed
Hit limit	Sets the maximum number of hits per hour permitted from a specific IP
Abuse limit	Sets that length of time in hours an IP address will be blocked out after exceeding the hit limit
Deny datasets	A comma delimited list of datasets that should not be accessible
Allow datasets	A comma delimited list of datasets that should be accessible

Level 3 users will have access to the parallel computing power of the LIS system, including an account on the LIS cluster and a web interface for submitting LIS jobs, as shown in Figure 14. The configuration parameters entered into the web form will be converted to LIS configuration files to control model runs. A LIS configuration file is submitted to the LIS scheduler which runs the job and places the output in a user-unique output directory in proper format for visualization. All the parameters will have default values.

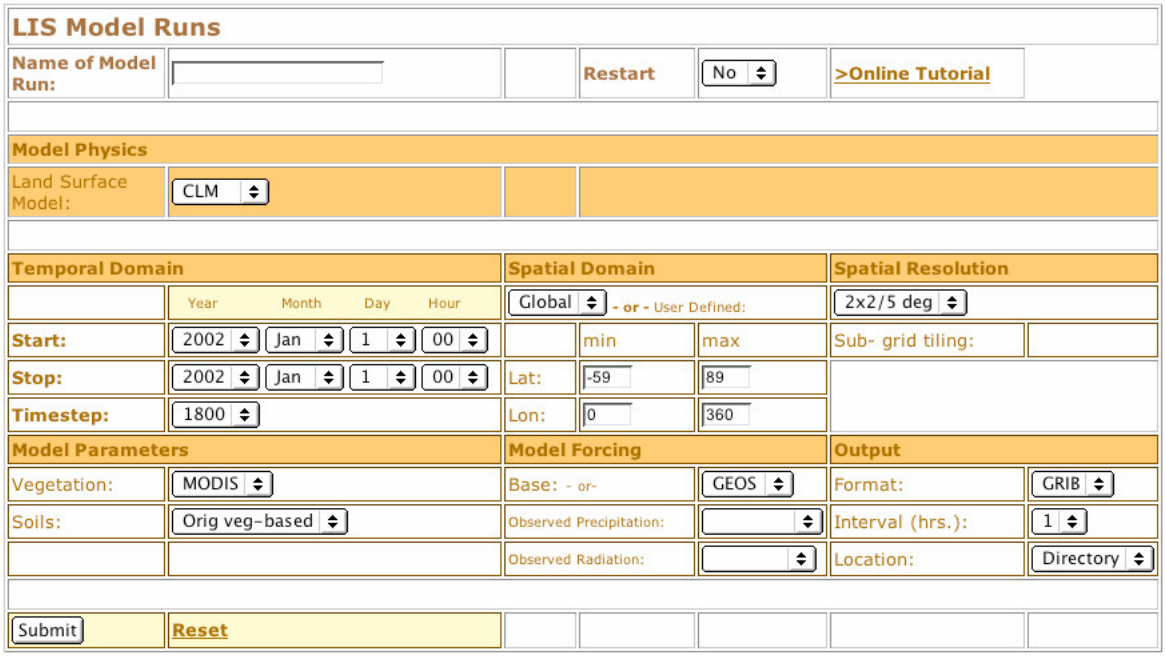

Figure 13: Sample design of LIS User Interface (Level 3)

References

ALMA: http://www.lmd.jussieu.fr/ALMA/

Atlas, R. M., and R. Lucchesi, File Specification for GEOS-DAS Gridded Output. Available online at: http://dao.gsfc.nasa.gov/DAO_docs/File_Spec_v4.3.html, 2000.

Chen, F., K. Mitchell, J. Schaake, Y. Xue, H. Pan, V. Koren, Y. Duan, M. Ek, and A. Betts, "Modeling of land-surface evaporation by four schemes and comparison with FIFE observations", *J. Geophys. Res.*, 101, D3, 7251-7268, 1996.

CLM: http://www.cgd.ucar.edu/tss/clm/

Collatz G. J., C. Grivet, J. T. Ball, and J. A. Berry, J. A. "Physiological and Environmental Regulation of Stomatal Conductance: Photosynthesis and Transpiration: A Model that includes a Laminar Boundary Layer", *Agric. For. Meteorol.* , 5, pp 107 -- 136, 1991.

Derber, J. C., D. F. Parrish, and S. J. Lord, "The new global operational analysis system at the National Meteorological Center", *Wea. And Forecasting,* 6, pp 538-547, 1991.

ESMF: http://www.esmf.ucar.edu/

GrADS-DODS server: http://grads.iges.org/grads/gds/

Hamill, T. M., R. P. d'Entremont, and J. T. Bunting, "A description of the Air Force real-time nephanalysis model", *Wea. Forecasting,* 7, pp 238-306, 1992.

Hofstee, H. P., J. J. Likkien, and J. L. A. Van De Snepscheut "A Distributed Implementation of a Task Pool". *Research Directions in High-Level Parallel Programming Languages*, pp 338--348, 1991.

Jarvis, P. G., " The interpretation of leaf water potential and stomatal conductance found in canopies in the field", *Phil. Trans. R. Soc.* London, Ser. B, 273, pp 593 – 610, 1976.

Kopp, T. J. and R. B. Kiess, "The Air Force Global Weather Central cloud analysis model", *AMS 15th Conf. on Weather Analysis and Forecasting,* Norfolk, VA, pp 220-222, 1996.

LDAS: http://ldas.gsfc.nasa.gov/

NOAH: http://www.emc.ncep.noaa.gov/mmb/gcp/noahlsm/README_2.2.htm

Pfaendtner, J., S. Bloom, D. Lamich, M. Seablom, M. Sienkiewicz, J. Stobbie, and A. da Silva, "Documentation of the Goddard Earth Observing System (GEOS) Data

Assimilation System – Version 1", *NASA Technical Memorandum* 104606, 4, pp 44, 1995.

Reynolds, C. A., T. J. Jackson, and W. J. Rawls, "Estimating available water content by linking the FAO Soil Map of the World with global soil profile databases and pedo-transfer functions" *American Geophysical Union, Fall Meeting, Eos Trans. AGU*, 80, 1999.

Richards, L. A., "Capillary conduction of liquids in porous media", *Physics,* 1, pp 318—333, 1931.

Rogers, E., T. L. Black, D. G. Deaven, G. J. DiMego, Q. Zhao, M. Baldwin, N. W. Junker, and Y. Lin, "Changes to the operational "early" eta analysis / forecast system at the National Centers for Environmental Prediction" *Wea. Forecasting*, 11, pp 391-413, 1996.

Shapiro, R. "A simple model for the calculation of the flux of direct and diffuse solar radiation through the atmosphere", AFGL-TR-87-0200, Air Force Geophysics Lab, Hanscom AFB, MA.

Turk, F. J., G. Rohaly, J. D. Hawkins, E. A. Smith, A. Grose, F. S. Marzano, A. Mugnai, and V. Levizzani, "Analysis and assimilation of rainfall from blended SSM/I, TRMM, and geostationary satellite data", *AMS 10th Conf. On Sat. Meteor. and Ocean.,* Long Beach, CA, 9-14 January, pp 66-69, 2000.

VIC: http://hydrology.princeton.edu/research/lis/index.html.